УДК 519.684.4
DOI: 10.15587/1729-4061.2019.171632

Дослідження паралелізм емпіричних моделей оптимальної складності за допомогою мережі петрі

М. І. Горбійчук, О.Т. Біла, Т. В. Гуменюк

Багато фізичних процеси і явища, з огляду на свою складність, не можуть бути описані аналітично. У таких випадках застосовують емпіричне моделювання. Для побудови емпіричних моделей оптимальної складності, яка має вигляд полінома заданого степеня, в роботі використаний метод, в основі якого лежить генетичний підхід. Реалізація розробленого методу вимагає багаторазового розв'язування системи лінійних алгебраїчних рівнянь. Розв'язування системи лінійних алгебраїчних рівнянь здійснюється шляхом приведення відповідної матриці, до верхньої діагональної формі з одиницями на головній діагоналі. Аналіз алгоритму приведення матриці до верхнього діагонального вигляду показав, що така процедура володіє внутрішнім паралелізмом. На основі створеної моделі обчислювального процесу у вигляді мережі Петрі розроблено стратегію побудови паралельного алгоритму для розв'язування системи лінійних алгебраїчних рівнянь. Суть стратегії в тому, що обчислення здійснюються на декількох паралельних процесорах. Одному з них присвоєні координуючі функції, і він названий майстром. Інші процесори - робітники - знаходяться в підпорядкуванні майстра. Поділ обсягу обчислень такий, що кількість рядків матриці, з якими оперує майстер, більша не менше ніж на одиницю, за відповідну кількість рядків, відведених робітникові. Для запропонованої стратегії оцінена ефективність паралельного алгоритму за критерієм сумарної кількості арифметичних операцій. Запропонована стратегія є складовою частиною процесу синтезу емпіричної моделі оптимальної складності на основі генетичних алгоритмів. Поділ обчислювального навантаження між паралельно працюючими процесорами (майстром і робочими) забезпечує прискорення обчислювального процесу в п'ять і більше разів.

Ключові слова: емпірична модель, генетичний алгоритм, паралелізм, мережа Петрі, число операцій

1. Введение

Если есть исчерпывающие данные об определенной системе (объекте), то можно получить модель, которая будет адекватно предопределять поведение такой системы при допущениях и при указанных параметрах среды, с которой система взаимодействует. Такие модели получили название детерминированных и используются для решения целого ряда задач в различных областях человеческой деятельности.

В большинстве случаев системы (объекты) являются достаточно сложными, из-за наличия многих компонентов, которые сложным образом

взаимодействуют между собой. Примером таких систем могут быть как технические, так и экологические объекты [1, 2]. Для описания взаимодействия между параметрами системы и внешней средой, непосредственно действующей на систему и вызывающей изменение ее состояния, используют эмпирическое моделирование. Математические модели, полученные в результате эмпирического моделирования, широко используются для решения таких задач как распознавание образов, прогнозирование, автоматическая классификация, оптимальное управление.

В работе [3] разработан метод синтеза моделей оптимальной сложности на принципах генетических алгоритмов. Реализация метода показала, что определение структуры и параметров модели требует значительных вычислительных затрат. Поэтому актуальной научной задачей является разработка и исследование параллельного алгоритма с использованием сети Петри, с целью ускорения вычислительного процесса.

2. Анализ литературных данных и постановка проблемы

В настоящее время для построения эмпирических моделей используются такие методы, как классический метод наименьших квадратов (МНК), байесовский подход, метод опорных векторов и непараметрического ядерного оценивания.

Согласно МНК, имея структуру модели, необходимо, используя наблюдения, как за входными, так и выходной переменными, определить параметры модели по методу наилучшего приближения, в качестве которого используется сумма квадратов отклонений экспериментальных данных от расчетных. Указанный критерий является внутренним [4] и его применение приводит к противоречивому результату: чем сложнее такая модель, тем она точнее. Как правило, на наблюдаемую выходную величину объекта накладывается помеха, которая искажает реально существующую функциональную зависимость между входами и выходом объекта.

Авторы работ [4, 5] для выбора наилучшей модели из заданного класса моделей (как правило, регрессионных) используют внешнее дополнение, которое представляет собой определенную часть из полученной выборки экспериментальных данных. Метод отбора наилучшего приближения по внешнему дополнению получил название индуктивный метод самоорганизации моделей.

Относительно задачи синтеза эмпирической модели применение внешнего критерия дает возможность однозначно выбрать структуру модели из заданного класса моделей.

Реализация индуктивного метода самоорганизации моделей требует значительных машинных затрат. Поэтому с целью уменьшения таких затрат в работе [3] предложен метод отбора структуры модели из заданного класса моделей, в основе которого лежат идеи генетических алгоритмов. При этом удалось значительно уменьшить затраты машинного времени, хотя такие затраты остаются еще достаточно большими.

В настоящее время возможности повышения эффективности вычислительных процессов, благодаря увеличению тактовой частоты процессоров, почти исчерпаны или экономически нецелесообразны. Одним из способов решения проблемы увеличения эффективности вычислительного процесса является его распараллеливание [6, 7].

Развитие параллельных вычислений привело к появлению целого ряда научных работ [7, 8, 9], где разработаны параллельные алгоритмы решения задач, требующих значительных затрат машинного времени. В качестве примеров можно назвать задачи цифровой фильтрации, оценки качества функционирования сложных динамических систем, решение систем линейных алгебраических уравнений с разреженными матрицами. В работе [10] дана оценка эффективности параллельных вычислительных процессов при решении перечисленных задач, исходя из условия, что процессоры загружены равномерно. Такая ситуация встречается очень редко. Как правило, распределения блоков матрицы между процессорами неравномерно. Таким образом, при синтезе эмпирических моделей оптимальной сложности нерешенной является задача разработка алгоритма и исследование его эффективности для случая, когда вычислительные мощности между процессорами распределены неравномерно.

3. Цель и задачи исследования

Целью исследования является разработка параллельного алгоритма синтеза эмпирических моделей оптимальной сложности, что даст возможность получить ускорение вычислительного процесса по сравнению с линейными вычислениями.

Для достижения цели были поставлены следующие задачи:

– алгоритмизировать задачи синтеза эмпирических моделей оптимальной сложности на принципах генетических алгоритмов;

– разработать стратегию взаимодействия параллельных процессоров с использованием сетей Петри;

– оценить эффективность разработанного параллельного алгоритма синтеза эмпирических моделей оптимальной сложности.

4. Эмпирическая полиномиальная модель оптимальной сложности на основе генетических алгоритмов

Метод построения эмпирических моделей на основе генетических алгоритмов, как и методы группового учета аргументов (МГУА), имеет в своей основе внешние критерии регулярности и смещения [5].

Использование внешних критериев предполагает, что выборка экспериментальных данных разбивается на две части – учебную N_A и проверочную. Учебное множество N_A используется для определения параметров модели, которая в большинстве случаев имеет полиномиальную структуру:

$$y = \sum_{i=0}^{M-1} a_i \prod_{j=1}^{n} x_j^{s_{ji}}, \qquad (1)$$

где n – количество независимых переменных (входные величины), от которых зависит значение выхода объекта исследования; $a_i, i = \overline{0, M-1}$ – коэффициенты модели (1); s_{ji} – степени аргументов, на которые накладывается ограничение:

$$\sum_{j=1}^{n} s_{ji} \le m, \ \forall i. \qquad (2)$$

Число членов полиномиальной модели (1) вычисляют по формуле [6]:

$$M = \frac{(m+n)!}{m!n!}. \qquad (3)$$

Отбор оптимальной относительно структуры эмпирической модели (1) осуществляется на проверочном множестве N_B по критерию регулярности [5]

$$\Delta^2(B) = \frac{\sum_{i=1}^{N_B} (Y^{(i)}(B) - y^{(i)}(B))^2}{\sum_{i=1}^{N_B} Y^{(i)}(B)^2} \qquad (4)$$

или смещения

$$\Delta^2(A,B) = \frac{\sum_{i=1}^{N} \left(y^{(i)}(A) - y^{(i)}(B) \right)^2}{\sum_{i=1}^{N} \left(Y^{(i)} \right)^2}, \qquad (5)$$

где $y^{(i)}(A)$ и $y^{(i)}(B)$, $i = \overline{1, N_A(N_B)}$ – значения выхода модели (1), вычисленных на множествах N_A и N_B; $Y^{(i)}(B), i = \overline{1, N_A}$ – экспериментальные значения выходной величины, отнесенные к множеству N_B; $Y^{(i)}, i = \overline{1, N}$ – выборка данных наблюдений за исходной величиной Y.

В отличие от алгоритмов МГУА, где базовые функции в виде регрессов включаются в модель (1) по переборным процедурам, в методе синтеза оптимальных по сложности эмпирических моделей на основе генетических алгоритмов (ГА-метод) отбор регрессов модели (1) осуществляется с использованием механизмов естественного отбора на основе критериев селекции (4) или (5). Такой подход предусматривает создание упорядоченной последовательности из единиц и нулей. Если на i-том месте находится единица

($a_i \neq 0$), то i-тый регресс включается в модель (1). В том случае, когда i-й регресс отбрасывается ($a_i=0$), то на i-том месте будет ноль. Такая упорядоченная последовательность, элементы которой единицы и нули, в теории генетических алгоритмов носит название хромосом, а ее атомарный элемент – это ген. Набор хромосом образует популяцию. Теперь задачу синтеза эмпирических моделей оптимальной сложности можно сформировать в терминах генетических алгоритмов. Последовательно отбирая «лучшие» хромосомы из всей популяции нужно найти ту, для которой функция приспособления (4) или (5) примет минимальное значения. Отобранная таким образом хромосома будет определять структуру эмпирической модели, которая будет оптимальной в классе моделей (1).

Формально алгоритм ГА-метода можно описать следующим образом:

$$GA = \langle I, CP, SA, SE, NP \rangle. \qquad (6)$$

Кортеж операторов (6) определяет последовательность их выполнения.

Оператор I случайным образом формирует начальную популяцию с I особей, каждая из которых является хромосомой длиной M.

На сегодняшний день не существует теоретических предпосылок выбора количества хромосом I в популяции [11–13]. При малом числе хромосом в популяции алгоритм быстро заканчивает свою работу и существует опасность, что функция приспособления не достигнет своего минимального значения. Увеличение числа хромосом в популяции влечет за собой увеличение машинных затрат на реализацию генетического алгоритма. Поэтому, значение I выбирают, исходя из интуиции исследователя и на основе машинных экспериментов.

Оператор CP оценивает приспособленность хромосомы в популяции. Для каждой хромосомы исчисляется критерий селекции (4) или (5) следующим образом. Исходную матрицу F, элементы которой регрессы при коэффициентах a_i, $i = \overline{0, M-1}$, вычисленные на множестве значений вектора входных переменных объекта, разбивают на две отдельные матрицы F_A и F_B размерами $N_A \times M$ и $N_B \times M$. Из матриц F_A и F_B изымается i-тый столбец, если на i-той позиции хромосомы находится ноль; если единица, то соответствующий столбец остается без изменений. В результате получаем матрицы $\tilde{F}_A$ и $\tilde{F}_B$, из которых изъято c столбцов (по количеству нулей в хромосоме).

Размер таких матриц – $N_A \times (M-c)$ и $N_B \times (M-c)$. На множестве точек N_A вычисляются ненулевые коэффициенты a_{Aj}, $j = \overline{0, M-c-1}$ модели (1) путем решения нормального уравнения Гаусса:

$$\tilde{M}_{F,A} \overline{a}_A = \tilde{F}_A^T \overline{Y}_A, \qquad (7)$$

где $\overline{a}_A = (a_{A0}, a_{A1}, \ldots, a_{A,M-c-1})^T$ – вектор ненулевых параметров модели, который ассоциирован с очередной хромосомой; $\tilde{M}_{F,A} = \tilde{F}_A^T \tilde{F}_A$ – матрица Фишера; $\overline{Y}_A = (Y^{(1)}, Y^{(2)}, \ldots, Y^{(N_A)})$ – вектор экспериментальных значений выхода объекта на множестве N_A.

В зависимости от того, какой критерий селекции (4) или (5) будет использоваться, по найденным коэффициентам a_{Aj}, $j = \overline{0, M-c-1}$ полиномиальной модели (1) на множестве точек N_A и N_B вычисляют значение

$$\overline{y}(A) = \tilde{F}_A \overline{a}_A, \tag{8}$$

или

$$\overline{y}(\mathbf{B}) = \tilde{F}_B \overline{a}_A. \tag{9}$$

Зная $\overline{y}(A)$ и $\overline{y}(\mathbf{B})$, по формуле (4) или (5) вычисляют значение функции приспособления $\Delta_j^2(A,B)$ или $\Delta_j^2(\mathbf{B})$, $j = \overline{1, I}$ для каждой хромосомы с изначальной популяции.

Оператор SA проверяет условия окончания работы алгоритма ГА-метода. Для каждой хромосомы со сложившейся популяции I вычисляют критерий селекции (4) или (5). Отбор лучшей хромосомы из популяции I, которая определит структуру оптимальной по сложности эмпирической полиномиальной модели, осуществляется по следующему правилу. Определяют

$$\Delta_m^2(\mathbf{B}) = \min_j \Delta_j^2(\mathbf{B}) \tag{10}$$

или

$$\Delta_m^2(A,B) = \min_j \Delta_j^2(A,B), \quad j = \overline{1, I}. \tag{11}$$

Если минимальное значение (10) или (11) критерия селекции (4) или (5) не превышает некоторого заданного значения, то происходит остановка вычислений. Остановка вычислений также может произойти, когда в результате выполнения алгоритма нет существенного уменьшения функции приспособления или в том случае, когда выполнено заданное число итераций.

После выполнения одной из трех условий из очередной популяции выбирается хромосома ch^*, для которой выполняется условие (10) или (11). Эта хромосома задает структуру модели оптимальной сложности и формирует матрицу F^* таким образом, что с исходной матрицы F изымаются столбцы, которые ассоциированы с нулевыми значениями соответствующих генов

хромосомы ch^*. Пересчет параметров модели (1) осуществляется на всем множестве точек с помощью МНК.

В том случае, когда не выполняется ни одно из трех перечисленных условий, происходит выполнение следующего оператора *SE*.

Оператор SE осуществляет селекцию хромосом. По рассчитанным значениям функции приспособления оператором *CP* осуществляется отбор из популяции *I* тех хромосом, которые будут участвовать в создании потомков для новой популяции. Такой отбор проводится по принципу естественного отбора, когда наибольшие шансы для создания новой популяции имеют хромосомы с лучшими значениями функции приспособления (4) или (5). Самым распространенным методами селекции является метод рулетки и турнирный метод [11]. Метод рулетки можно применять тогда, когда функция приспособления положительная, что делает его пригодным только для задач максимизации. Турнирный метод можно использовать как в задачах максимизации, так и в задачах минимизации.

При турнирной селекции все хромосомы разбиваются на подгруппы по z_0 особей. С каждой подгруппы выбирается лучшая хромосома согласно критерию селекции (4) или (5). В результате получаем новую популяцию хромосом, которая образует родительский пул $I(k)$. Количество особей I в популяции остается неизменным. Подгруппы могут иметь произвольный размер, но чаще всего популяцию разделяют на подгруппы по 2–3 особи в каждой.

Количество хромосом, которое генерируется оператором *I*, должно быть кратным числу особей z_0 в подгруппах. В данном алгоритме использован турнирный метод.

Оператор NP формирует новую популяцию потомков из отобранных хромосом оператором *SE*. Формирование новой популяции хромосом осуществляется с помощью процессов скрещивания и мутации. Вероятность скрещивания P_h выбирается из интервала [0,5; 1], а вероятность мутации находится в пределах [0; 0,1].

Процессу скрещивания подлежит каждая отобранная хромосома оператором *SE*, которая принадлежит к родительскому пулу. Для этого из популяции особей случайным образом выбирается пара хромосом. Генерируется случайное число P_c из интервала [0,5; 1] и, если его значение не превышает P_h, то над парой хромосом осуществляется скрещивание. В противном случае пара хромосом остается неизменной. Если имеет место скрещивание хромосом, то для каждой пары разыгрывается позиция гена (локус), которая определяет точку скрещивания. Хромосома каждого из родителей имеет M генов, и точка скрещивания L_c – это натуральное число меньше M. Поэтому фиксация точки скрещивания сводится к случайному выбору целого числа L_c из интервала $[1, M–1]$.

Процесс скрещивания приводит к тому, что из пары родителей образуется новая пара потомков. Первый потомок в паре хромосом, который на позициях от 1 до L_c состоит из генов первого родителя, а на позициях L_c+1 до M – из генов второго родителя. Второй потомок в паре хромосом на позициях от 1 до

L_c состоит из генов второго родителя, а на позициях L_c+1 до M – из генов первого родителя.

Вероятность мутации P_m можно задать путем выбора случайного числа с интервала [0; 1] для каждого гена. Мутации подлежат те гены (путем замены единицы на ноль и, наоборот), для которых разыгранное число окажется меньше или равным P_m. Мутация может осуществляться как над пулом родителей, так и над пулом потомков.

После выполнения оператора *NP* осуществляется переход к оператору *CP*.

Реализация алгоритма ГА-метода показала [2] значительный выигрыш во времени по сравнению с переборным МГУА алгоритмом, хотя с увеличением размерности задачи растут затраты машинного времени на его реализацию.

5. Исследование алгоритма синтеза эмпирической модели оптимальной сложности с помощью сети Петри

Анализ алгоритма ГА-метода показал [14], что значению критерия приспособления (4) или (5), который вычисляется для каждой хромосомы, соответствует параллельная форма, которая, наряду с другими операциями, включает в себя операцию решения нормального уравнения Гаусса (7). Размерность последнего определяется степенью полинома (1) и количеством входных переменных.

Так, в работе [15] синтезирована эмпирическая модель (1), в которой $n = 7$ и m=4. В соответствии с формулой (3) максимальное число параметров модели (1) M=330. Такое значительное число параметров приводит к тому, что решение уравнения (7) (особенно на завершающем этапе работы алгоритма) потребует дополнительных затрат машинного времени.

Уменьшить затраты времени на реализацию алгоритма ГА-метода можно, если в каждой параллельной форме дополнительно распараллелить алгоритм Гаусса решения системы линейных алгебраических уравнений (7).

Нормальное уравнение (7) запишем в следующей форме:

$$\tilde{A}\overline{a}_A = \tilde{b}, \tag{12}$$

где $\tilde{A} = \tilde{M}_{F,\mathrm{A}}$; $\tilde{b} = \tilde{F}_A^T \overline{Y}_A$.

Уравнения (12) будем решать, используя метод гауссова исключения, приведя расширенную матрицу $A_e = [\tilde{A} \mid \tilde{b}]$ к верхнему диагональному виду согласно следующим формулам [16]:

$$\tilde{a}_{i\bullet}^{(i)} = \frac{\tilde{a}_{i\bullet}^{(i-1)}}{\tilde{a}_{ii}^{(i-1)}}, \ i = \overline{1, M-c-1}; \tag{13}$$

$$\tilde{a}_{k\bullet}^{(i)} = \tilde{a}_{k\bullet}^{(i-1)} - \tilde{a}_{i\bullet}^{(i)} \tilde{a}_{ki}^{(i-1)}, \ k = \overline{i+1, M-c}, \tag{14}$$

где $\tilde{a}_{i\bullet}^{(i)}$, $\tilde{a}_{k\bullet}^{(i)}$, i-тый и k-тый строки расширенной матрицы A_e; $\tilde{a}_{i\bullet}^{(0)} = \tilde{a}_{i\bullet}$, $i = \overline{1, M-c-1}$; $\tilde{a}_{k\bullet}^{(0)} = \tilde{a}_{k\bullet}$, $k = \overline{i+1, M-c}$. В формуле (13) $\tilde{a}_{ii}^{(i-1)}$ носит название ведущего элемента.

В вычислительном процессе цикл по индексу k является внутренним по отношению к внешнему циклу по индексу i. Результатом применения итерационных процедур (13) и (14) является верхняя диагональная матрица с единицами на главной диагонали:

$$U = \begin{bmatrix} 1 & u_{12} & u_{13} & \cdots & u_{1,M-c} & u_{1,M-c+1} \\ 0 & 1 & u_{23} & \cdots & u_{2,M-c} & u_{2,M-c+1} \\ 0 & 0 & 1 & \cdots & u_{3,M-c} & u_{3,M-c+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 & u_{M-c,M-c+1} \end{bmatrix},$$

где $u_{ij} = \tilde{a}_{ij}^{(i)}$, $i = \overline{1, M-c}$, $j = \overline{i+1, M-c+1}$.

Таким образом, уравнение (12) преобразуется в эквивалентную систему линейных алгебраических уравнений:

$$\tilde{U}\,\bar{a}_A = \bar{b}_A, \tag{15}$$

где $\tilde{U}$ – квадратная матрица размером $M–c$, образованная с матрицы путем изъятия последнего столбца; $b_{A,i=}u_{i,M-c+1}$, $i = \overline{1, M-c}$.

Уравнение (12) решается методом обратной прогонки, начиная с последнего уравнения. Очевидно, что

$$a_{A,M-c} = b_{A,M-c}. \tag{16}$$

Другие значения $a_{A,i}$ вычисляются по итерационной процедуре:

$$a_{A,i} = b_{A,i} - \sum_{j=i+1}^{M-c} u_{ij} a_{A,j}, \quad i = \overline{M-c-1, 1}. \tag{17}$$

В отличие от классического метода Гаусса [17], в процессе приведения системы уравнений (12) к виду (15) отсутствует операция деления на u_{ii}. для вычисления $a_{A,i}$. Как известно, на выполнение операции деления тратится больше машинного времени по сравнению с другими алгебраическими операциями. Поэтому вычисления параметров модели (4) по рекуррентным соотношениям (16) и (17) дает ощутимую экономию машинного времени по сравнению с классическим методом Гаусса.

Сказанное верно и для *LU*-метода, в соответствии с которым матрица $\tilde{A}$ представлена в виде произведения двух матриц L и U_R. Первая из них – нижняя диагональная с единицами на главной диагонали, а вторая – верхняя диагональная матрица. Затем прямой и обратной прогонкой решаются треугольные системы $L\overline{\gamma} = \overline{b}_A$, $U_R\overline{a}_A = \overline{\gamma}$.

Поскольку в процессе решения уравнения (12) на приведение матрицы A_e к верхнему диагональному виду расходуется большая часть машинного времени, то для уменьшения таких затрат целесообразно распараллелить соответствующий алгоритм.

Допустим, что есть q параллельных процессов. Один из них, назовем его мастером, выполняет функцию управления параллельным вычислительным процессом. Другие q–1 процессы (рабочие) подчинены мастеру.

Суть алгоритма распараллеливания в том, на каждом шагу вычислений происходит разделение матрицы, которая находится в рабочем пространстве мастера на q подматриц с последующим вычислением значений их элементов по формулам (13) и (14). В результате подматрица A, которая находится в рабочем пространстве мастера, приводится к верхнему диагональному виду с единицами на главной диагонали. Одновременно с мастером рабочие пересчитывают элементы своих подматриц по формуле (14) и после окончания очередного цикла вычислений пересылают их мастеру. Полученные от рабочих подматрицы мастер объединяет в одну матрицу, которую назовем объединенной матрицей и обозначим как $A_U^{(i)}$, $i = 1, 2, \ldots$.

Выберем такую процедуру разделения объединенной матрицы $A_U^{(i)}$, которая находится в рабочем пространстве мастера, на отдельные подматрицы. Допустим, что на каждом этапе вычислений строки объединенной матрицы $A_U^{(i)}$, $i = 1, 2, \ldots$ делятся на число q без остатка.

Перед началом первого этапа вычислений матрицу $A_U^{(1)} = A_e$ разделим на q равных частей. Тогда все q подматриц будут иметь одинаковое количество строк

$$\tilde{n}_1 = z / q, \qquad (18)$$

где $z=A_e$.

После окончания первого этапа вычислений количество строк объединенной матрицы $A_U^{(1)}$ уменьшилась на величину $\tilde{n}_1$. Поэтому в рабочем пространстве мастера будет объединенная $A_U^{(2)}$ матрица со следующим количеством строк: $z_1 = z - \tilde{n}_1$.

Учитывая значение $\tilde{n}_1$, которое определяется формулой (18), получим

$$z_1 = z - \frac{z}{q} = z\frac{q-1}{q}.$$

В начале второго этапа вычислений мастер делит объединенную матрицу $A_U^{(2)}$ снова на q равных частей, что приводит к следующему результату: $\tilde{n}_2 = \frac{z_1}{q}$.

С учетом значения z_1, получим

$$\tilde{n}_2 = z\frac{q-1}{q^2}. \tag{19}$$

Результатом окончания второго цикла вычислений будет объединенная матрица $A_U^{(2)}$ с z_2 строками $z_2 = z - \tilde{n}_1 - \tilde{n}_2$.

Учитывая значения $\tilde{n}_1$ и $\tilde{n}_2$, которые определены формулами (18) и (19), получим $z_2 = z\frac{(q-1)^2}{q^2}$.

В начале третьего этапа вычислений объединенную матрицу $A_U^{(3)}$ с количеством строк z_2 мастер снова делит на q равных частей. В результате такого деления в рабочих пространствах мастера и рабочих будут находиться подматрицы с количеством строк:

$$\tilde{n}_3 = z\frac{(q-1)^2}{q^3}. \tag{20}$$

После выполнения третьего этапа вычислений получим матрицу $A_U^{(4)}$ с количеством строк: $z_4 = z - \tilde{n}_1 - \tilde{n}_2 - \tilde{n}_3$. Принимая во внимание $\tilde{n}_1$, $\tilde{n}_2$ и $\tilde{n}_3$ получим $z_4 = \frac{z}{q^3}(q-1)^3$.

Если теперь матрицу $A_U^{(4)}$ разделить на q равных частей, то

$$\tilde{n}_4 = \frac{z}{q^4}(q-1)^3. \tag{21}$$

Продолжая процесс деления объединенной матрицы после окончания r-того этапа вычислений получим матрицу $A_U^{(r)}$ с таким количеством строк: $z_r = \frac{z}{q^r}(q-1)^r$.

Для продолжения работы алгоритма на r+1 цикле мастер делит матрицу $A_U^{(r)}$ на q равных частей, так что

$$\tilde{n}_r = \frac{z}{q^r}(q-1)^{r-1},\ r = \overline{1,\ N_z - 1}, \tag{22}$$

где N_z – общее количество этапов вычислений.

В общем случае значения $\tilde{n}_r$, $r = \overline{1, N_z - 1}$ ненатуральные числа.

Для каждого из q–1 рабочих мощность каждой части (количество строк) объединенной матрицы $A_U^{(r)}$, $r = \overline{1, N_z - 1}$ будем определять, как целую часть числа $\tilde{n}_r$, $r = \overline{1, N_z - 1}$. Итак, $n_{w,r} = \left[\frac{z}{q^r} (q-1)^{r-1} \right]$, $r = \overline{1, N_z - 1}$, где [·] – целая часть числа.

Количество строк $n_{f,r}$ подматрицы, которая находится в рабочем пространстве мастера, будем вычислять как разность между количеством строк z_r , $r = \overline{1, N_z - 1}$ объединенной матрицы $A_U^{(r)}$ и количеством строк, которые суммарно делегированы q–1 рабочим

$$n_{f,r} = z_r - n_{w,r}(q-1), \; r = \overline{1, N_z - 1}. \tag{23}$$

Если в процессе такого деления объединенных матриц $A_U^{(r)}$ окажется, что $n_{f,r} < n_{w,r}$, то $n_{w,r}$ постепенно уменьшается на единицу до достижения условия

$$n_{f,r} \geq n_{w,r}. \tag{24}$$

При этом после каждого такого уменьшения значение $n_{f,r}$ пересчитывается по формуле (23).

С учетом требования (24) количество строк подматрицы, что находится в рабочем пространстве каждого рабочего, будет вычисляться по формуле:

$$n_{w,r} = \left[\frac{z}{q^r} (q-1)^{r-1} \right] - v_r, \; r = \overline{1, N_z - 1}, \tag{25}$$

где v_r – число единиц, на которое уменьшилась величина $n_{w,r}$ после выполнения условия (24).

В результате выбранной стратегии разделения объединенных матриц в рабочем пространстве мастера будут размещаться подматрицы с количеством строк $n_{f,r}$, а каждый рабочий получит подматрицу количеством строк $n_{w,r}$. Значения $n_{f,r}$ и $n_{w,r}$ вычисляются по формулам (23) и (25) при условии выполнения неравенства (24).

Разделение мастером объединенных матриц продолжается до тех пор, пока размер очередного слоя, подлежащего отправке, будет не больше количества процессов

$$z_s \leq q. \tag{26}$$

Тогда $n_{f,s=}z_s$, а в рабочем пространстве каждого рабочего $n_{w,s}$=0.

Условие (26) определяет окончание процесса приведения исходной матрицы A_e к верхнему диагональному виду с единицами на главной диагонали.

После выполнения условия (26) мастер приводит оставшуюся подматрицу к верхнему прямоугольному виду с единицами на главной диагонали. Завершающий этап – объединение всех подматриц, которые были сохранены в рабочем пространстве мастера.

Оценим количество вычислительных этапов N_z, необходимых для реализации алгоритма приведения матрицы A_e к верхнему диагональному виду с единицами на главной диагонали.

Допустим, что на последнем этапе работы вычислительного алгоритма условие (26) будет таким:

$$n_{f,s} = q. \tag{27}$$

Мощность z_r объединенной матрицы $A_U^{(r)}$ на каждом этапе вычислений уменьшается на величину $n_{f,j}$. Поэтому на r-том этапе вычислений получим

$$z_r = z - n_{f1} - n_{f2} - \cdots - n_{f,r} = z - \sum_{j=1}^{r} n_{f,j}. \tag{28}$$

После окончания N_z–1 этапа вычислений мощность объединенной матрицы $A_U^{(N_z)}$ будет $n_{f,s}$. Исходя из условия (27) и формулы (28), получаем $n_{f,s} = z - \sum_{j=1}^{N_z-1} n_{f,j}$.

Поскольку имеет место допущение (27), то

$$z - \sum_{j=1}^{N_z-1} n_{f,j} = q. \tag{29}$$

В соответствии с выбранной стратегией разделения объединенной матрицы на q подматриц между мощностями подматриц мастера и каждого из рабочих справедливо соотношение (25), исходя из которого можно утверждать, что

$$n_{f,j} - n_{w,j} = R_j,\ j = \overline{1, N_z - 1}, \tag{30}$$

где $0 \le R_j \le q$ – некоторое целое число.

Мощности подматриц, которые находятся в рабочем пространстве рабочих, представим как целую часть числа $\frac{z}{q^j}(q-1)^{j-1}$, учитывая при этом, что должно выполняться условие (25)

$$n_{w,j} = \frac{z}{q^j}(q-1)^{j-1} - r_j - v_j, \tag{31}$$

где $0 \le r_j < 1$ – отброшенная дробная часть числа $\tilde{n}_j$.

С формул (30) и (31) следует, что

$$n_{f,j} = \frac{z}{q^j}(q-1)^{j-1} + \delta_j, \tag{32}$$

где

$$\delta_j = R_j - (r_j + v_j), \ j = \overline{1, N_z - 1}.$$

Подставляя полученное значение $n_{f,j}$ в формулу (29), получим

$$z\left(1 - \sum_{j=1}^{N_z-1} \frac{1}{q^j}(q-1)^{j-1}\right) - \Delta = q,$$

где $\Delta = \sum_{j=1}^{N_z-1} \delta_j$.

Выражение, что находится в скобках, представим в следующей форме:

$$1 - \frac{1}{q}\sum_{j=1}^{N_z-1}\left(\frac{q-1}{q}\right)^{j-1}.$$

Величина $\sum_{j=1}^{N_z-1}\left(\frac{q-1}{q}\right)^{j-1}$ является суммой геометрической прогрессии, в которой первый член единица, а знаменатель $(q-1)/q$; количество членов геометрической прогрессии – N_z-1. Вычислив сумму геометрической прогрессии, получим $1 - \frac{1}{q}\sum_{j=1}^{N_z-1}\left(\frac{q-1}{q}\right)^{j-1} = \left(\frac{q-1}{q}\right)^{N_z-1}$.

Вычисленное значение суммы, позволяет формулу (32) записать в следующем виде:

$$z\left(\frac{q-1}{q}\right)^{N_z-1} - \Delta = q. \qquad (33)$$

Из соотношения (33) определим количество этапов, которые необходимо выполнить, чтобы привести матрицу A_e к верхнему диагональному виду

$$N_z = 1 + \ln\left(\frac{q+\Delta}{z}\right) \Big/ \ln\left(\frac{q-1}{q}\right). \qquad (34)$$

Значение N_z учитывает также и последний этап, который будет иметь место после выполнения условия (26). На завершающем этапе мастер последнюю часть матрицы A_e приводит к верхнему диагональному виду, а рабочие уже закончили свою работу.

Формула (34) определяет количество этапов необходимых для окончания работы алгоритма с определенной погрешностью. Погрешность возникает из-за того, что работа алгоритма заканчивается тогда, когда на последнем этапе вычислений в рабочем пространстве мастера будет объединена матрица мощностью q. Фактически, алгоритм прекращает свою работу после выполнения условия (26).

На рис. 1 показан результат определения количества этапов вычислений, необходимых для приведения матрицы A_e к верхнему диагональному виду. Были использованы такие входные данные: размеры квадратных матриц $A_e(i), i=\overline{1,8}$ сформированы как вектор – $\bar{z} = [10000\ 8000\ 6000\ 4000\ 2000\ 1000\ 500\ 300]$; количество процессоров q=6. Для каждой матрицы $A_e(i), i=\overline{1,8}$ исчислялось число этапов по формуле (34) и фиксировалось количество этапов как результат выполнения условия (26).

Наглядное представление о распределении мощностей подматриц мастера и рабочих при выполнении i, $(i=\overline{1,8})$-го этапа вычислений дает рис. 2, который показывает, что всегда выполняется условие (26), при этом использовались следующие исходные данные: z=300; q=6.

Для оценки эффективности вычислительного процесса, прежде всего, нужны рабочие модели, которые адекватно характеризуют структурные и динамические свойства процесса и отражают параллелизм алгоритма приведения матрицы A_e к верхнему диагональному виду.

При моделировании параллельных вычислительных процессов широкое применение получили сети Петри [18, 19], которые адекватно отражают взаимодействие основных частей процесса и информационный обмен между ними.

На рис. 3 изображена сеть Петри, которая является моделью вычислительного процесса приведения матрицы A_e к верхнему диагональному виду.

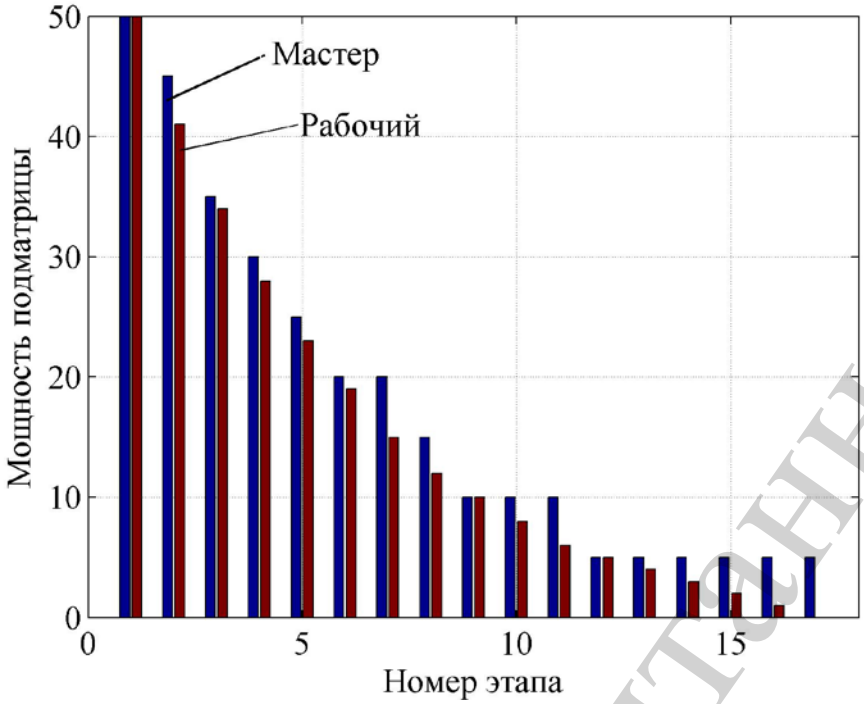

Рис. 1. Результат вычислительного эксперимента

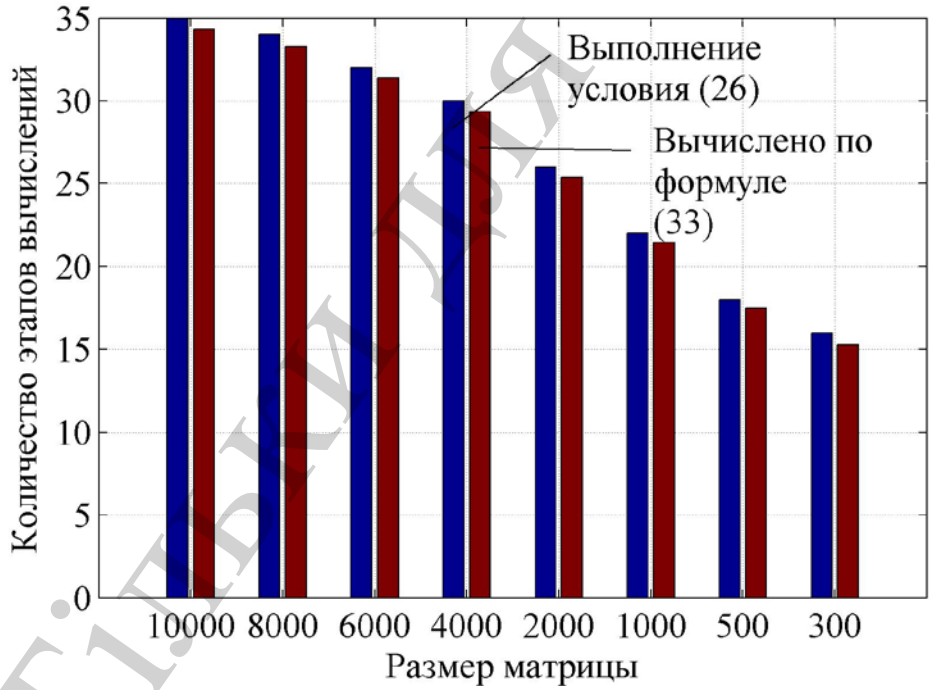

Рис. 2. Распределение мощностей подматриц мастера и рабочих по этапам вычислений

Позиция p_s содержит маркер, который активизирует переход t_s, вызывая выполнение первого шага вычислений. Позиция $p_A^{(0)}$ идентифицируется как процесс рассылки мастером рабочим подматриц $A_0^{(i)}$, $i = \overline{1, q-1}$. Подматрицы

$A_0^{(i)}, i = \overline{1, q-1}$ имеют одинаковые размеры, и количество их строк определяется формулой (25). В рабочем пространстве мастера остается подматрица $A_0^{(0)}$ размером $n_{f1}\times(M - c+1)$, где количество строк подматрицы вычисляется по формуле (23).

В дальнейшем вычисления осуществляются, так же, как и на первом шаге. После k-го шага вычислений в рабочем пространстве мастера будет находиться подматрица $A_k^{(0)}$ с количеством строк, вычисляемых по формуле (23), а подматрицы $A_k^{(i)}, i = \overline{1, q-1}$ будут иметь размер $n_{w1}\times(M–c+1)$, в которых количество нулевых столбцов $P_{w,k} = \sum_{j=1}^{k} n_{f,k}$. После вычисления мастером компонентов вектора $\tilde{a}_k$. и рассылки их значений рабочим, активизируются переходы $t_i^k, i = \overline{0, q-1}$. Трансформация подматриц $A_k^{(i)}, i = \overline{1, q-1}$ на k-ом шаге вычислений моделируется позициями $p_k^{(i)}, i = \overline{1, q-1}$.

Завершение первого этапа вычислений происходит при условии, что мастер привел свою подматрицу $A_m^{(0)}$ к верхнему диагональному виду. Тогда происходит слияние подматриц $A_m^{(i)}$ в объединенную матрицу $A_U^{(1)} = \bigcup_{i=1}^{q-1} A_m^{(i)}$, что отображается позицией p_{sum}.

Позиции p_D, p_c і p_f формируют условия завершения работы алгоритма приведения матрицы A_e к верхнему диагональному виду.

Позиция p_e имеет постоянный маркер. Переход t_{sz} сработает в том случае, когда количество строк подматрицы $A_m^{(s)}$, которая приведена к верхнему диагональному виду (в позиции p_{sz} появится маркер), не будет превышать значения q. После этого все подматрицы, которые находятся в рабочем пространстве мастера, объединяются в матрицу U (позиция p_U) и процесс приведения матрицы к верхнему диагональному виду заканчивается.

В случае выполнения условия (26) активизируется переход t_F. В позиции p_c появляется маркер, что приводит к срабатыванию перехода t_{sum} и, как следствие, маркер переходит в позицию p_s.

В процессе приведения исходной матрицы A_e к верхнему диагональному виду мастер и рабочие работают параллельно, при этом имеет место соотношение (25). Время, затраченное на решение исходной задачи, будет определяться количеством арифметических операций, которые выполняет мастер и рабочие, как на каждом этапе вычислений, так и в целом до завершения решения задачи.

Для случая, когда распределения вычислительной нагрузки между мастером и рабочими было произвольным, задача нахождения количества арифметических операций на каждом шаге работы алгоритма решена в [20].

Сначала вычислим количество операций на каждом шагу вычислений, выполняемых мастером, допуская, что количество строк подматрицы, расположенные в рабочем пространстве мастера, вычисляется по формуле (23).

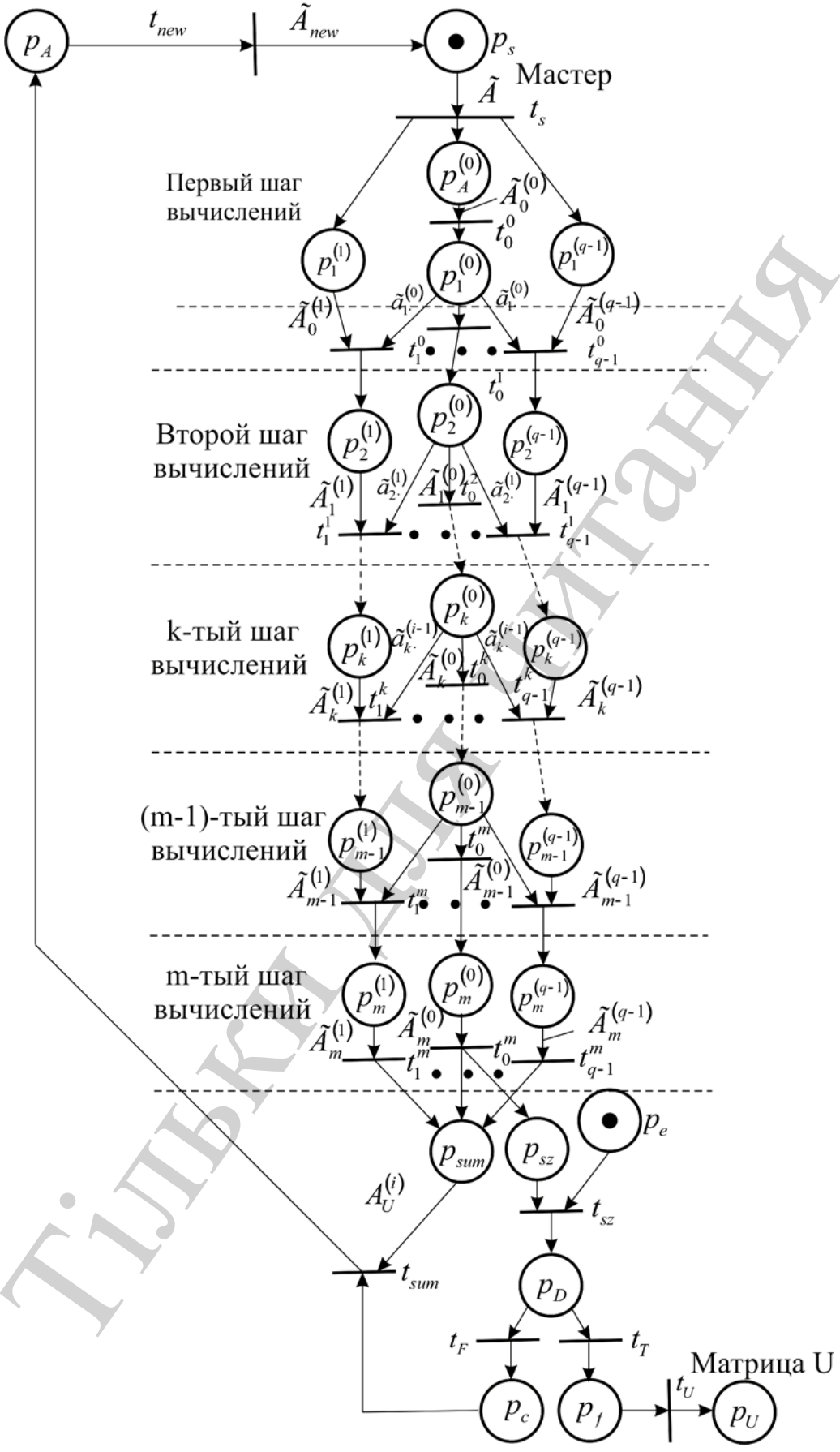

Рис. 3. Моделирование алгоритма приведения матрицы к верхнему диагональному виду с помощью сети Петри

Выполнение мастером операции нормирования первой строки (рис. 3) подматрицы $\tilde{A}_0^{(0)}$ в соответствии с формулой (13) требует $z=M-c$ операций деления (диагональным элементам подматрицы $\tilde{A}_0^{(0)}$ присваивается значение единицы). На перечисление элементов остальных n_{f1} строк подматрицы $\tilde{A}_0^{(0)}$ по формуле (14) будет затрачено $(n_{f1}-1)z$ операций умножения и $(n_{f1}-1)z$ операций вычитания. Общее количество операций, выполненных мастером на первом этапе, составит $N_{f1}=(2n_{f1}-1)z$.

В результате выполнения первого шага вычислений начальная подматрица $\tilde{A}_0^{(0)}$ модифицируется до подматрицы $\tilde{A}_1^{(0)}$, в которой первый столбец содержит нулевые элементы, кроме первого, который равен единице. Подматрицы $\tilde{A}_0^{(i)}, i=\overline{1,q-1}$, расположенные в рабочих пространствах рабочих, будут трансформированы в подматрицы $\tilde{A}_1^{(i)}, i=\overline{1,q-1}$ с нулевыми первыми столбцами.

На втором шаге вычислений (рис. 3) мастер нормирует вторую строку подматрицы $\tilde{A}_1^{(0)}$, затратив $z-1$ операций деления. Другие строки матрицы $\tilde{A}_1^{(0)}$ пересчитываются по формуле (14). При этом будет выполнено $(n_{f1}-2)(z-1)$ операций умножения и такое же количество операций вычитания. Итак, на втором шаге вычислений мастер выполнит $N_{f2}=2(n_{f1}-2)(z-1)+(z-1)=2(n_{f1}-3)(z-1)$ арифметических операций. При этом будет получена модифицированная матрица $\tilde{A}_2^{(0)}$ (рис. 3).

На третьем шаге вычислений мастер нормирует третью строку матрицы $\tilde{A}_2^{(0)}$ (рис. 3), выполнив $z-2$ операцию деления. Начиная с четвертой строчки, мастер перечисляет все элементы подматрицы $\tilde{A}_2^{(0)}$ по формуле (14), тратя такое количество операций деления, умножения и вычитания: $N_{f3}=3(n_{w1}-5)(z-2)$. В результате в рабочем пространстве мастера будет модифицирована матрица $\tilde{A}_3^{(0)}$ (рис. 3).

Обобщая полученные результаты, можно утверждать, что на k-ом шаге вычислений мастер выполнит такое количество операций умножения, деления и вычитания:

$$N_{f1,k}=2((n_{f1}-k)+0,5)(z-k+1). \qquad (35)$$

Первый этап параллельных вычислений заканчивается тогда, когда выполнится условие $k=n_{f1}$, то есть в формуле (35) $k=\overline{1,n_{f1}}$.

Таким образом, после первого этапа вычислений в рабочем пространстве мастера получаем верхнюю прямоугольную матрицу $\tilde{A}_{n_{f1}}^{(0)}$ с единицами на главной диагонали. Размер такой матрицы $n_{f1}\times(z+1)$.

После объединения мастером q–1 подматриц в его памяти будут храниться подматрицы $\tilde{A}^{(0)}_{n_{f1}}$ и объединенная матрица $A^{(1)}_U$ (рис. 3), в которой есть n_{f1} нулевых столбцов. Размер последней – $(z–n_{f1})\times(z–n_{f1}+1)$.

Объединенную матрицу $A^{(1)}_U$ мастер делит на q слоев в соответствии с формулами (23) и (25). В рабочем пространстве мастера будет находиться подматрицы $\tilde{A}^{(0)}_{\Sigma,1}$ со n_{f2} строками, а рабочие получат подматрицы $\tilde{A}^{(1)}_{\Sigma,1}$, каждая из которых будет иметь n_{w2} строк, которые вычисляются по формуле (25) при r=2.

Количество операций деления, умножения и вычитания, которые выполняет мастер на втором шаге работы алгоритма, вычислим по формуле (35), учитывая, что количество ненулевых столбцов подматрицы, расположенной в рабочих пространстве мастера, равно $z–n_{f1}+1$. Это означает, что в формуле (35) необходимо z заменить на $z–n_{f1}$, а n_{f1} на n_{f2}. В результате такой замены получим $N_{f1,k=}2((n_{f2}–k)+0,5)(z–n_{f1}–k+1)$.

Второй этап вычислений закончится при выполнении условия $k=n_{f2}$.

После окончания второго этапа вычислений в памяти мастера будет сохранена прямоугольная матрица $A^{(U)}_{\Sigma,2} = A^{(0)}_{n_{f1}} \bigcup A^{(0)}_{\Sigma,1}$ размером $(n_{f1}+n_{f2})(z+1)$, на главной диагонали которой будут размещены единицы.

В начале третьего цикла вычислений мастер объединяет q–1 подматрицу в одну матрицу $A^{(2)}_U$ размером $(z–n_{f1}–n_{f2})\times(z–n_{f1}–n_{f2}+1)$, которая не содержит нулевых элементов. Полученную матрицу мастер делит на q подматриц. Первая подматрица $A^{(0)}_{\Sigma,3}$, которая имеет n_{f3} строк, остается в рабочем пространстве мастера, а другие подматрицы $A^{(1)}_{\Sigma,3}$ со n_{w3} строками отправляются q–1 рабочим.

По аналогии со вторым этапом, количество операций деления, умножения и вычитания, что выполняет мастер на третьем этапе, вычислим по формуле (35), заменив z на $z–n_{f1}–n_{f2}$. При этом необходимо учесть, что мощность подматрицы $A^{(0)}_{\Sigma,3}$ составляет n_{f3} строк. Итак,

$$N_{f3,k} = 2((n_{f3} - k) + 0,5)(z - n_{f1} - n_{f2} - k + 1). \qquad (36)$$

Третий этап заканчивается при условии, что $k=n_{f3}$.

Пусть выполнено r этапов вычислений. В результате в памяти мастера будет размещена прямоугольная матрица $A^{(0)}_{\Sigma,r}$ размером $\left(\sum_{j=1}^{r-1} n_{f\,j}\right)\times(z+1)$, на главной диагонали которой будут размещены единицы. В рабочем пространстве мастера будет находиться объединенная матрица $A^{(r)}_U$ размером $(z-\sum_{j=1}^{r-1} n_{f\,j})\times(z-\sum_{j=1}^{r-1} n_{f\,j}+1)$ с ненулевыми элементами, которую мастер делит на q слоев. В результате получим подматрицу $A^{(0)}_{\Sigma,r}$ мощностью $n_{f,r}$, которая

находится в рабочем пространстве мастера. Каждая из подматриц рабочих $A^{(1)}_{\Sigma,r}$ имеет мощность $n_{w,r}$.

Опираясь на результаты формулы (36), можем утверждать, что на k-том шаге r-того этапа вычислений мастер выполнит такое количество операций умножения, деления и вычитания:

$$N_{fr,k} = 2((n_{f,r} - k) + 0,5)(z - \sum_{j=1}^{r-1} n_{f,j} - k + 1). \quad (37)$$

Окончанием r-того этапа является выполнение условия $k=n_{f,r}$.

Если в формуле (37) учесть значение $n_{f,j}$ по формуле (32), то получим

$$N_{fr,k} = 2((n_{f,r} - k) + 0,5)(z\left(1 - \frac{1}{q}\sum_{j=1}^{r-1}\left(\frac{q-1}{q}\right)^{j-1}\right) - \Delta_{r-1} - k + 1),$$

где $\Delta_{r-1} = \sum_{j=1}^{r-1} \delta_j$.

Вычислив сумму $\sum_{j=1}^{r-1}\left(\frac{q-1}{q}\right)^{j-1}$ по формуле геометрической прогрессии, находим

$$N_{fr,k} = 2((n_{f,r} - k) + 0,5)(z\left(\frac{q-1}{q}\right)^{r-1} - \Delta_{r-1} - k + 1). \quad (38)$$

Теперь вычислим количество операций деления, умножения и вычитания, которые мастером выполняются в течение r-того этапа вычислений

$$N_{M,r} = \sum_{k=1}^{n_{f,r}} N_{fr,k}, \ r = \overline{1, N_z - 1}. \quad (39)$$

Если учесть операции на пересылку $\tilde{a}_{ij}^{(\nu)}$ элемента на каждом k-потому шаге вычислений, то максимальное количество параллельных арифметических операций, которая выполняется на r-потому этапе будет такой:

$$N_r = N_{M,r} + n_{f,r}, \ r = \overline{1, N_z - 1}. \quad (40)$$

Вычислим значения $N_{M,r}$. Если учесть формулу (38), то выражение (39) примет следующий вид:

$$N_{M,r} = 2\sum_{k=1}^{n_{f,r}}\left(\left(n_{f,r}-k\right)+0{,}5\right)\left(z\left(\frac{q-1}{q}\right)^{r-1}-\Delta_{r-1}-k+1\right).$$

Принимая во внимание [21], что

$$\sum_{k=1}^{n_{f,r}}k=\frac{n_{f,r}}{2}(n_{f,r}+1)\ \text{i}\ \sum_{k=1}^{n_{f,r}}k^2=\frac{n_{f,r}}{6}(n_{f,r}+1)(2n_{f,r}+1),$$

получим

$$\begin{aligned}N_{M,r} &= 2n_{f,r}((n_{f,r}+0{,}5)(z\alpha^{r-1}-\Delta_{r-1}-\frac{1}{2}(n_{f,r}-1))-\\ &-(n_{f,r}+1)(\frac{1}{2}(z\alpha^{r-1}-\Delta_{r-1})-\frac{1}{3}(n_{f,r}-1))),\end{aligned} \tag{41}$$

где $\alpha=(q-1)/q$.

Для вычисления общего числа арифметических операций, выполняемых мастером, необходимо взять сумму по всем операциям по каждому из r этапов вычислений: $N_{sum}=\sum_{r=1}^{N_z-1}N_{M,r}+\sum_{r=1}^{N_z-1}n_{\text{f},r}$. Очевидно, что $\sum_{r=1}^{N_z-1}n_{\text{f},r}=z-n_{f,s}$. Поэтому

$$N_{sum}=\sum_{r=1}^{N_z-1}N_{M,r}+z-n_{f,s}, \tag{42}$$

где $n_{f,s}$ – мощность последней объединенной матрицы в рабочем пространстве мастера, не подлежащей дальнейшему разделению между рабочими.

На последнем этапе вычислений, когда выполнено условие (26), мастер подматрицу $\tilde{A}^{(0)}_{\Sigma,s}$ размером $\left(z-\sum_{j=1}^{r-1}n_{f,j}\right)\times\left(z-\sum_{j=1}^{r-1}n_{f,j}+1\right)$ приводит к верхнему диагональному виду в соответствии с формулами (13) и (14).

Вычислим количество операций умножения, деления и вычитания, которые выполняются мастером после последнего вычисления. Введем следующие обозначения: $z_\alpha=z-\sum_{j=1}^{r-1}n_{f,j}$. Матрицу $A^{(s)}_U$ можно рассматривать как объединенную квадратную матрицу размером z_α, к которой присоединен столбец свободных членов системы алгебраических линейных уравнений. Приведение такой матрицы к верхнему диагональному виду с единицами на главной диагонали требует [22]

$$N_{f,s}=\frac{4z_\alpha^3+3z_\alpha^2-z_\alpha}{6}, \tag{43}$$

операций деления, умножения и вычитания.

С учетом формулы (43) количество операций, которые выполняет мастер в параллельном вычислении, будет следующим:

$$N_{sum} = \sum_{r=1}^{N_z-1} N_{M,r} + \frac{4z_\alpha^3 + 3z_\alpha^2 - z_\alpha}{6} + z - n_{f,s}. \qquad (44)$$

Формула (44) позволяет проанализировать эффективность параллельного алгоритма по сравнению с последовательным алгоритмом приведения квадратной матрицы к верхнему диагональному виду с единицами на главной диагонали.

При приведении диагональной матрицы к верхнему диагональному виду непосредственно по формулам (13) и (14) необходимо затратить количество арифметических операций, которые вычисляется по формуле (43) после замены z_α на z.

Для оценки эффективности разработанного алгоритма приведения квадратной матрицы к верхнему диагональному виду было вычислено количество арифметических операций по формулам (41) и (44) и количество операций N_s по формуле (43) (после замены z_α на z). Эффективность алгоритма оценивалось как отношение $K_e=N_s/N_{sum}$.

Результат такой оценки отражает рис. 4. Были взяты следующие исходные данные:

$$\overline{z} = [10000\ 8000\ 6000\ 4000\ 2000\ 1000\ 500\ 300];$$

количество процессоров q=6. Анализ полученных результатов показывает, что применение параллельного алгоритма позволяет более чем в девять раз сократить количество арифметических операций необходимых для приведения исходной матрицы A_e к верхнему диагональному виду. Следует отметить, что эффективность разработанного алгоритма незначительно падает с уменьшением размера матрицы A_e.

Все N_r операций в r-том цикле вычислений выполняются параллельно. Допустим, что на выполнение всех операций умножения, деления и вычитания на r-том этапе вычислений тратится τ_r единиц времени, а на операции пересылки – $\tau_{t,r}$ единиц времени. Тогда общие затраты времени на реализацию параллельного алгоритма приведения матрицы A_e к верхней диагонального вида следует вычислять по формуле:

$$T = \sum_{r=1}^{N_z-1} \left(N_{M,r}\tau_r + \tau_{t,r} n_{f,r} \right) + T_f, \qquad (45)$$

где T_f – затраты времени мастером на приведение своей подматрицы на последнем шаге вычислений в верхней диагональной форме.

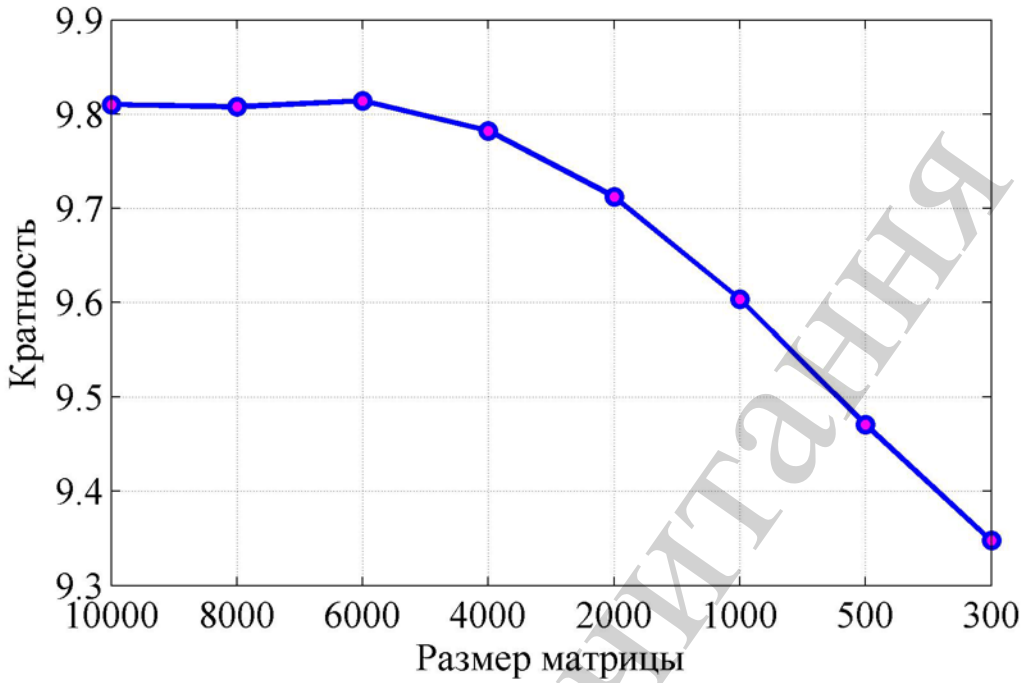

Рис. 4. Зависимость коэффициента K_e от размера матрицы A_e

С учетом значения $N_{f,s}$ формулу (45) запишем в следующем виде: $T = \sum_{r=1}^{N_z-1}\left(N_{M,r}\tau_r + \tau_{t,r}n_{f,r}\right) + \tau_f N_{f,s}$, где τ_f – затраты времени на выполнение арифметических операций на завершающем этапе вычислений.

После приведения матрицы A_e к верхнему диагональному виду с единицами па главной диагонали в обратном порядке определяются параметры модели (1) по формулам

$$a_{A,z} = u_{z,z+1},\ a_{A,i} = u_{i,z+1} - \sum_{j=i+1}^{z} u_{ij}a_{A,j},\ i = \overline{z-1,1}. \qquad (46)$$

Проанализируем алгоритм (46) решения системы линейных алгебраических уравнений с треугольной матрицей U методом обратной подстановки. Физический смысл задачи синтеза оптимальной структуры эмпирической модели такой, что U не вырожденная матрица.

Формула (46) не определяет алгоритм однозначно, так как не определен порядок нахождения суммы. Рассмотрим последовательную операцию определения суммы, находящейся в правой части соотношения (46). Соответствующая рекуррентная процедура будет следующей:

$$a_{A,z}^{(0)} = a_{A,z} = u_{z,z+1},$$

$$a_{A,z-i}^{(j)} = a_{A,z-i}^{(j-1)} - u_{z-i,z-j+1} a_{A,z-j+1}^{(j-1)},$$
$$a_{A,z-i}^{(0)} = u_{z-i,z+1},\ a_{A,z-i} = a_{A,z-i}^{(i)},\ i = \overline{1, z-1},\ j = \overline{1, i}. \quad (47)$$

Процедура нахождения решений уравнения (15) по формулам (47) обладает внутренним параллелизмом. Поэтому для экономии машинного времени синтезируем соответствующий параллельный алгоритм.

Распределение нагрузок между мастером и рабочими будем осуществлять по принципу, который имел место при приведении матрицы A_e к верхнему диагональному виду.

На первом этапе вычислений предпоследний z столбец матрицы U, который имеет z–1 отличных от нуля элементов (не считая элемента u_{zz}=1), разделим равномерно между q процессорами, так что мастер будет иметь $h_{f,1}$ элементов, а рабочие по $h_{w,1}$ элементов. По формуле (47) вычислим значения $a_{A,z-i}^{(1)}$, где $i = \overline{1, h_{f,1}}$ для мастера; $i = \overline{h_{f,1}+1, \mathrm{h}_{w,k}^{(1)}}$ для k=1 и $i = \overline{h_{w,k}^{(1)}+1, \mathrm{h}_{w,k+1}^{(1)}}$, $k = \overline{1, q-2}$ для рабочих. Рабочие пересылают свои значения $a_{A,z-i}^{(1)}$ мастеру, где формируется множество значений $a_{A,z-1}$ и $a_{A,z-i}^{(1)}$, $i = \overline{2, z-1}$, которые хранятся в рабочей области мастера.

На втором шаге z–2 элементы z–1 столбца разделим равномерно между q процессорами. В результате такого деления мастер получит $h_{f,2}$ элементов, а рабочие – $h_{w,k}^{(2)}$, элементов. После этого мастер, включая и самого себя, рассылает соответствующее количество элементов $a_{A,z-1}$ и $i = \overline{2, z-1}$, каждому процессору. Дальше по формуле (47) вычисляются значения $a_{A,z-i-1}^{(2)}$, где $i = \overline{1, h_{f,2}}$ для мастера; $i = \overline{h_{w,k}^{(2)}+1, \mathrm{h}_{w,k+1}^{(2)}}$, $k = \overline{1, q-2}$ для процессоров, при этом $a_{A,z-2} = a_{A,z-2}^{(2)}$. Завершается второй шаг формированием в рабочей области мастера множества значений $a_{A,z-2}$ и $a_{A,z-i}^{(2)}$, $i = \overline{3, z-1}$.

Дальнейшие шаги вычислений происходят по приведенной схеме и на r - том шаге мастер разделяет z–r элементы z–r+1 столбца между q процессорами следующим образом: мастер будет иметь $h_{j,r}$ элементов, а рабочие по $h_{w,k}^{(r)}$, $k = \overline{1, q-1}$ элементов. По формуле (47) вычисляются значения $a_{A,z-i-r+1}^{(r)}$, где $i = \overline{1, h_{f,r}}$ для мастера; $i = \overline{h_{w,k}^{(r)}+1, h_{w,k+1}^{(r)}}$, $k = \overline{1, q-1}$ для рабочих. Как результат вычислений в рабочей области мастера будет находиться множество значений $a_{A,z-r}$ и $a_{A,z-i}^{(r)}$, $i = \overline{r+1, z-1}$.

Процесс решения алгебраического уравнения (15) по методу обратного хода с использованием параллельного алгоритма завершается последним этапом вычислений при выполнении условия z–r=q. Тогда мастер вычисляет все значения $a_{A,z-i}$, где $i = \overline{r, z-1}$по формуле (47).

В общем случае размер каждого столбца матрицы U может оказаться не кратным количеству процессоров. Поэтому, как раньше, количество элементов,

которые будут находиться в рабочем пространстве рабочих, определим как целую часть числа

$$h_{w,k}^{(i)} = \left[\frac{z-i}{q} \right], \; i = \overline{1, r}, \; k = \overline{1, \; q-1}. \qquad (48)$$

Поскольку каждый рабочий на *i*-том шаге вычислений получает одинаковое количество элементов, то $h_{w,1}^{(i)} = h_{w,2}^{(i)} = \cdots = h_{w,q-1}^{(i)} = h_{w,i}$.

Количество элементов *z*–*i*+1 столбца, которые получает мастер на *i*-том шаге, будем вычислять по формуле:

$$h_{f,i} = z - i - h_{w,i}(q-1), \; i = \overline{1, \; z-1}. \qquad (49)$$

После выполнения условия *z*–*i*=*q* значения $h_{w,j}, \; j = \overline{1, \; q-1}$ обнуляются, а значения $h_{f,j}$ следует вычислять по следующей рекуррентной процедуре:

$$h_{f,j+1} = h_{f,j} - 1, \; j = \overline{z-q, \; z-1}, \qquad (50)$$

где $h_{f,z-q} = q$.

Покажем, что выбранный способ распределения вычислительной нагрузки между мастером и рабочими всегда обеспечивает выполнение неравенства

$$h_{f,i} \geq h_{w,i}. \qquad (51)$$

Значение величины $h_{w,i}$, которое вычисляется по формуле (48), представим в следующем виде:

$$h_{w,i} = \frac{z-i}{q} - \rho_i, \; i = \overline{1, r}, \qquad (52)$$

где $0 \leq \rho_i < 1$ – отброшена дробная часть числа $\frac{z-i}{q}$.

Подставляя полученное значение $h_{w,i}$ в (49), будем иметь $h_{f,i} = \frac{z-i}{q} + (q-1)\rho_i$.

Поскольку, $h_{f,i} - h_{w,i} = q\rho_i \geq 0$ то всегда имеет место соотношение (51).

Следовательно, распределение количества элементов между мастером и рабочими *z*–*i*+1 столбца осуществляется в соответствии с формулами (48) и (49) при этом всегда выполняется условие (50).

Вычислим количество операций умножения и вычитания, которые осуществляется в результате реализации параллельного алгоритма вычислений по формуле (47). На *r*-том шаге вычислений мастер выполнит $N_0^{(r)} = 2h_{f,r}$, а каждый *i*-й рабочий $N_i^{(r)} = 2h_{w,r}$ операций умножения и вычитания.

Поскольку параллельные вычисления ведутся по столбцам матрицы *U*, $r = \overline{1,z}$, и имеет место соотношение (51), то общее число операций при реализации параллельного алгоритма $N = \sum_{r=1}^{z-q} N_0^{(r)} + N_f$, где N_f – число операций умножения и вычитания на последних шагах вычислений.

При переходе от одного шага вычислений к другому число элементов очередного столбца уменьшается на единицу, поэтому на определенном этапе вычислений будет иметь место условие *z–i=q*. Учитывая условие завершения процесса вычислений, а так же формулу (51), приходим к выводу, что

$$N = 2\sum_{r=1}^{z-q} h_{f,r} + N_f. \qquad (53)$$

Теперь вычислим величину N_f. Завершающие шаги вычислений начинаются с условия $h_{f,s}=q$. Дальнейшие вычисления параметров модели выполняет мастер, вычисляя параметры модели согласно формуле (47). Поэтому общее количество операций умножения и вычитания на последнем этапе вычислений будет таким: $N_f = 2\sum_{i=0}^{q-1}(q-i) = q(q+1)$. Если учесть значение N_f, то формула (53) примет следующий вид: $N = 2\sum_{r=1}^{z-q} h_{f,r} + q(q+1)$.

Поскольку между нагрузками мастера и каждым рабочим существует соотношение (51), то

$$h_{f,i} - h_{w,i} = q\rho_i. \qquad (54)$$

С уравнения (54) следует, что $h_{f,i} = h_{w,i} + q\rho_i$. С учетом формулы (52), значение $h_{f,i} = \frac{z-i}{q} + (q-1)\rho_i$, $i = \overline{1,\ z-q}$.

Зная $h_{f,i}$, можем вычислить значение *N* в соответствии с формулой (54) $N = 2\sum_{r=1}^{z-q}\frac{z-r}{q} + \Lambda + q(q+1)$, где $\Lambda = (q-1)\sum_{r=1`}^{z-q}\rho_r$.

Вычислив значение $\sum_{r=1}^{z-q}(z-r)$, приходим к выводу, что

$$N = \frac{z-q}{q}(z+q-1) + q(q+1) + \Lambda.$$

Реализация последовательного алгоритма вычисления коэффициентов математической модели (1), после того как получена матрица U, требует [22] $\tilde{N} = z(z-1)$ операций умножения и вычитания.

Рис. 5 дает представление об эффективности параллельного алгоритма вычисления коэффициентов эмпирической модели (1) по формуле (47) по сравнению с последовательным алгоритмом.

Эффективность параллельного алгоритма оценивалась величиной кратности $K_s = \tilde{N}/N$. Анализ полученных результатов показывает, что с уменьшением размерности матрицы A_e эффективность параллельного алгоритма падает почти по экспоненте. Например, при z=100 и q=6 кратность K_s=5,3. Поскольку в алгоритме синтеза эмпирических моделей оптимальных по структуре на основе генетических алгоритмов процедуру решения системы уравнений необходимо повторять M–c раз, то даже при относительно малых размерах матрицы A_e выигрыш в вычислительном времени будет заметным.

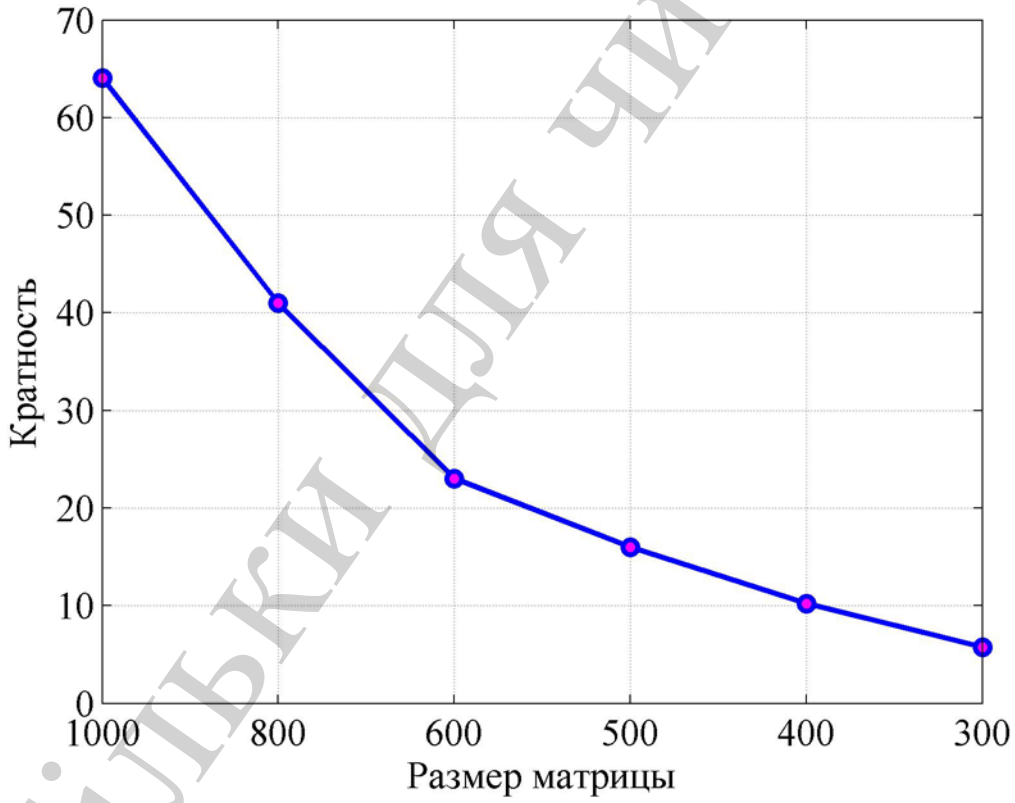

Рис. 5. Эффективность параллельного алгоритма при q=6

Суммарное количество арифметических операций, которая выполняется при реализации параллельного алгоритма, будет определяться количеством операций на приведение матрицы $\tilde{A}$ к верхнему треугольному виду и количеству операций на решение уравнения (15).

6. Обсуждение результатов исследования параллельного алгоритма с использованием сети Петри

Построение эмпирических моделей оптимальной сложности на принципах генетических алгоритмов требует многократного решения системы линейных алгебраических уравнений с последующим вычислением выхода модели. Такая процедура требует значительных затрат времени. Поэтому возникла необходимость в разработке алгоритма параллельных вычислений, которые в значительной степени ускоряют процесс синтеза эмпирической модели оптимальной сложности. Ранее поставленная задача решалась при условии, что вычислительные нагрузки между процессами распределены равномерно, что редко встречается на практике.

Предложена новая процедура разделения вычислительной нагрузки между координирующим процессором (мастером) и остальными процессорами (рабочими), когда вычислительная нагрузка мастера несколько больше вычислительной нагрузки каждого рабочего. Для неравномерной нагрузки между параллельными вычислительными процессами произведена оценка ускорения вычислительного процесса по сравнению с обычным алгоритмом.

Эффективность распараллеленного алгоритма увеличивается с ростом размерности задачи. В действительности эффективность параллельного алгоритма будет несколько ниже теоретической вследствие не учета операций пересылки и прерываний.

Для оценки реальной эффективности предложенного параллельного алгоритма синтеза моделей оптимальной сложности необходимо дополнительно провести целый ряд вычислительных экспериментов.

7. Выводы

1. В отличие от индуктивного метода самоорганизации моделей, авторы предлагают синтезировать эмпирическую модель в виде полинома заданной степени. Каждому коэффициенту модели ставится в соответствие «1» или «0» (1 – коэффициент присутствует в модели; 0 – коэффициент исключен из модели). В результате структура модели кодируется последовательностью, состоящей из нулей и единиц, названой хромосомой. Генерируется множество хромосом – популяция, количество хромосом в которой определяется структурой полного полинома. Над полученной популяцией производится последовательность операций, основные из которых скрещивание и мутация. Использование функции приспособления позволяет отобрать наиболее «приспособленную» хромосому, которая и определяет структуру эмпирической модели оптимальной сложности.

2. Реализация метода синтеза эмпирической модели оптимальной сложности показала, что, по сравнению с индуктивным методом самоорганизации, предложенный метод требует меньших затрат машинного времени. Для сложных моделей с большим числом переменных и высокой степенью полинома ($m \geq 3$) затраты машинного времени могут быть значительны (до одного часа и более). Для выявления путей уменьшения таких затрат разработана модель вычислительного процесса в виде сети Петри.

Анализ модели показал, что процесс синтеза эмпирической модели оптимальной сложности обладает внутренним параллелизмом.

3. Разработана стратегия распределения вычислительных операций между мастером и рабочими, при которой все рабочие загружены равномерно, а нагрузка мастера всегда больше, чем нагрузка рабочих. Такая стратегия разделения позволила в явной форме определить необходимое количество арифметических операций для реализации параллельного алгоритма. Показано, что разработанная стратегия разделения нагрузки между параллельно работающими процессорами (мастером и рабочими) обеспечивает ускорение вычислительного процесса в пять и более раз.

Литература

1. Горбийчук М. И., Щупак И. В., Осколип Т. Метод синтеза эмпирических моделей с учетом погрешностей измерений // Методы и приборы контроля качества. 2011. № 2 (27). С. 67–76.
2. Горбийчук М. И., Шуфнарович М. А. Метод построения математических моделей сложных процессов на основе генетических алгоритмов // Искусственный интеллект. 2010. № 4. С. 50–57.
3. Метод синтезу емпіричних моделей на засадах генетичних алгоритмів / Горбійчук М. І., Когутяк М. І., Василенко О. Б., Щупак І. В. // Розвідка та розробка нафтових і газових родовищ. 2009. № 4. С. 72–79.
4. Степашко В. С., Булгакова А. С. Обобщенный итерационный алгоритм метода группового учета аргументов // Управляющие системы и машины. 2013. № 2. С. 5–17.
5. Gupta S., Bhardwaj S., Bhatia P. K. A reminiscent study of nature inspired computation // International Journal of Advances in Engineering & Technology. 2011. Vol. 1, Issue 2. P. 117–125.
6. Voevodin V., Antonov A., Popova N. Studying the Structure of Parallel Algorithms as a Key Element of High-Performance Computing Education // Lecture Notes in Computer Science. 2019. P. 199–210. doi: https://doi.org/10.1007/978-3-030-10549-5_16
7. Ortega J. M. Introduction to parallel and vector solution of linear systems. Springer, 1988. doi: https://doi.org/10.1007/978-1-4899-2112-3
8. Dvukhglavov D. E., Kulynych V. E. Development of software solution for building route of a orders group delivery in presence of time constraints // Bulletin of National Technical University "KhPI". Series: System Analysis, Control and Information Technologies. 2017. Issue 55. P. 64–71. doi: https://doi.org/10.20998/2079-0023.2017.55.11
9. Параллельные алгоритмы решения задач вычислительной математики / Химич А. Н., Молчанов И. Н., Попов А. В., Чистякова Т. В., Яковлев М. Ф. К.: Наукова думка, 2008. 248 с.
10. Khimich A. N., Popov A. V., Polyankoa V. V. Algorithms of parallel computations for linear algebra problems with irregularly structured matrices // Cybernetics and Systems Analysis. 2011. Vol. 47, Issue 6. P. 973–985. doi: https://doi.org/10.1007/s10559-011-9377-4

11. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. М.: Горячая линия-Телеком, 2004. 452 с.

12. Богатырев М. Ю. Инварианты и симметрии в генетических алгоритмах. URL: http://www.raai.org/conference/cai-08/files/cai-08_paper_287.pdf

13. Гладков Л. А., Курейчик В. В., Курейчик В. М. Генетические алгоритмы. 2-е изд., испр и доп. М.: ФИЗМАТЛИТ, 2006. 320 с.

14. Горбійчук М. І., Медведчук В. М., Пашковський Б. В. Паралелізм алгоритму синтезу моделей оптимальної складності на засадах генетичних алгоритмів // Восточно-Европейский журнал передовых технологий. 2014. Т. 4, № 2 (70). С. 42–48. doi: https://doi.org/10.15587/1729-4061.2014.26305

15. Горбійчук М. І., Шуфнарович М. А. Метод синтезу математичних моделей коливних процесів з некратними частотами // Науковий вісник Івано-Франківського національного технічного університету нафти і газу. 2010. № 1. С. 105–112.

16. Luszczek P. Parallel Programming in MATLAB // The International Journal of High Performance Computing Applications. 2009. Vol. 23, Issue 3. P. 277–283. doi: https://doi.org/10.1177/1094342009106194

17. Вержбицкий В. М. Основы численных методов: учеб. М.: Высшая школа, 2002. 840 с.

18. Патерсон Дж. Теория сетей Петри и моделирование систем. М.: Мир, 2000. 263 с.

19. Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В. Моделирование параллельных процессов. Сети Петри: курс для системных архитекторов, программистов, системных аналитиков, проектировщиков сложных систем управления. Санкт-Петербург: Профессиональная литература, 2014. 398 с.

20. Gorbiychuk M. I., Medvedchuk V. M., Lazoriv A. N. Analysis of Parallel Algorithm of Empirical Models Synthesis on Principles of Genetic Algorithms // Journal of Automation and Information Sciences. 2016. Vol. 48, Issue 2. P. 54–73. doi: https://doi.org/10.1615/jautomatinfscien.v48.i2.60

21. Корн Г., Корн Т. Справочник по математике для научных работников и инженеров. М.: Наука, 1978. 832 с.

22. Волков Е. А. Численные методы: учеб. пос. 2-е изд., исп. М.: Наука, 1987. 248 с.